461 гр. Тарасова Е. С.
Научный руководитель ст. пр. Полозов В. С.
Рецензент Зеленчук И. В.

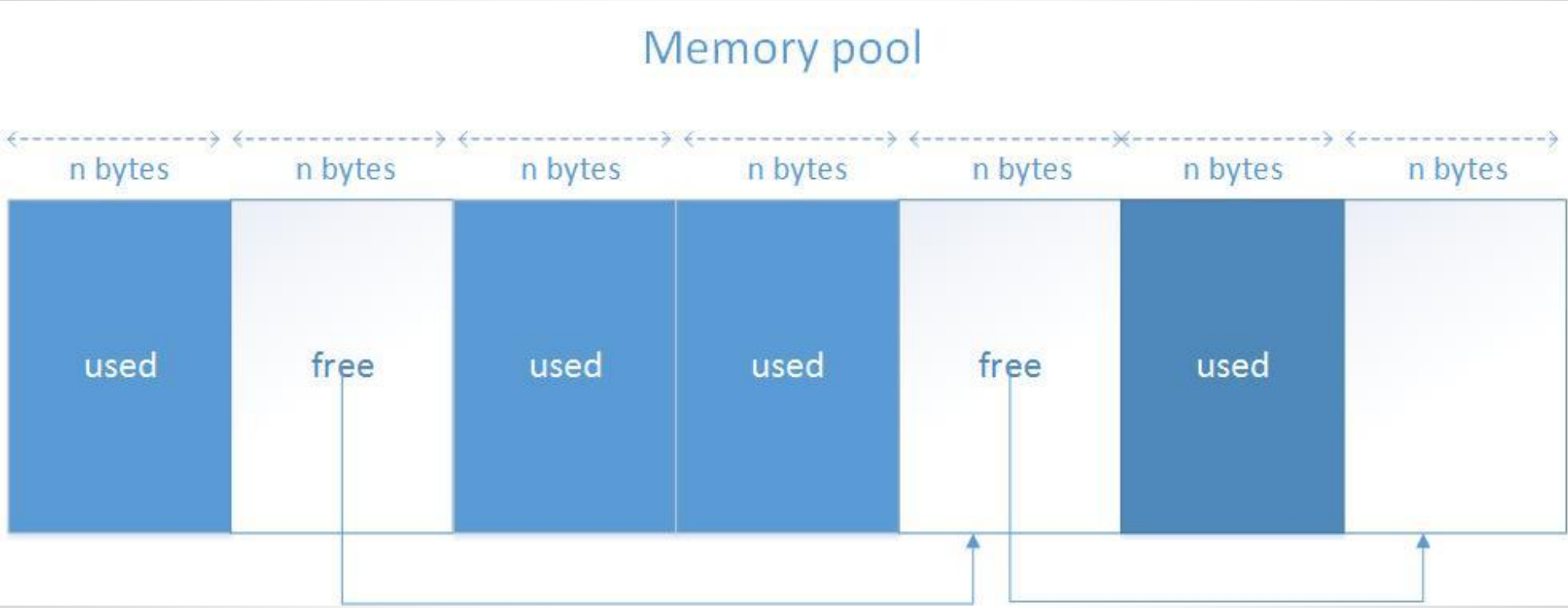
Бакалаврская работа

Оптимизация выделения памяти на основе статического анализа программы в инфраструктуре Clang

Задача оптимизации выделения памяти

- В некоторых случаях примерно 30% времени работы программы уходит на организацию памяти
- Существует возможность повысить эффективность программы путем повышения эффективности функций работы с памятью

Pool



- Последовательный участок памяти
- Блоки одинакового размера
- Ограниченная область применения

Постановка задачи

- Провести сравнительный анализ эффективности алгоритмов выделения памяти
  - Программы с использованием malloc/free,
  - Программы с ипользованием pool
  - Провести измерение времени и анализ результатов
- Реализовать прототип инструмента, оптимизирующего выделение памяти в программе
- Провести анализ работы инструмента

Анализ эффективности алгоритмов выделения памяти

- Windows 8, Visual Studio 2012

	malloc/free	pool
Вставка и удаление 700000 элементов в контейнер std::vector	0.073 с	0.9 с
Вставка и удаление 700000 элементов в контейнер std::set	0.065 с	0.066 с
Вставка и удаление 700000 элементов в контейнер std::list	379.861 с	2.778 с
Выделение и освобождение памяти для 700000 элементов размера 64 байта	14.415 с	0.017 с

Результаты произведенных измерений

Поиск	Замена

std::list<int> x;
for (...)
{
x.push_back(some_int);
}
	std::list<int, pool_alloc> x;
	for (...)
	{
	x.push_back(some_int);
	}

for (…)
{
…
a = std::malloc(size);
…
std::free(a); ...
}
	boost::pool<> pool(size);
	for (…)
	{
	...
	a = pool.malloc();
	...
	pool.free(a); ...
	}

Результаты произведенных измерений

Поиск	Замена

int ** p = malloc (...);
for (…) {... p[i] = (int *) std::malloc(size); ...}
for (…) { ... std::free(p[i]); …}
	boost::pool<> pool(size);
	int ** p = malloc (...);
	for (…) {... p[i] =(int *) (pool.malloc()); …}
	for (…) {... pool.free(p[i]); …}

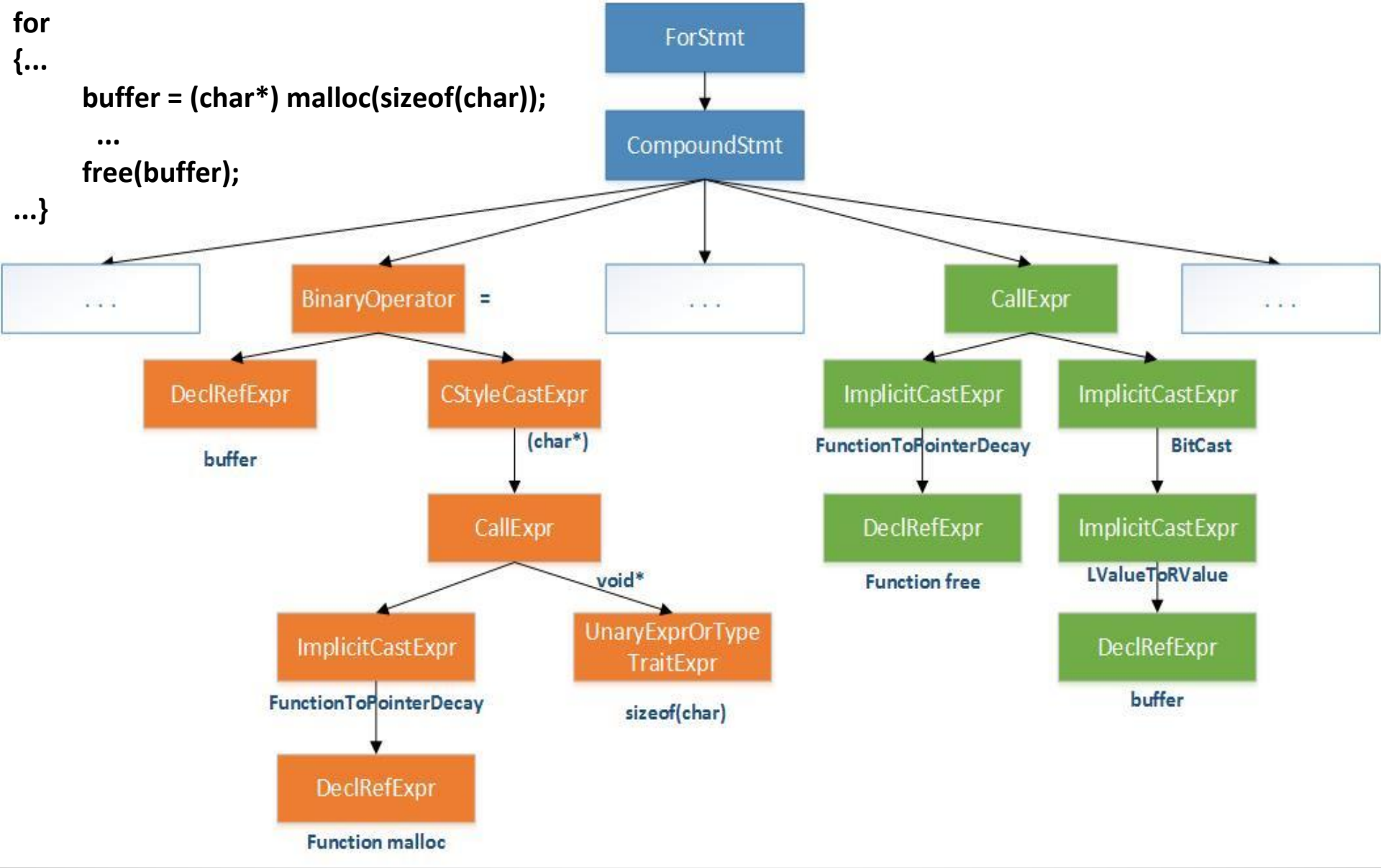
type1 function1()
{
for (...)
{
function2();
}
}
 	
type2 function2()
{...
a = std::malloc(size);
...}
	boost::pool<> pool(size);
	type1 function1()
	{
	for (...)
	{
	function2();
	}
	}
	type2 function2()
	{...
	a = pool.malloc());
	...}

Реализация

- Программа с использованием библиотеки Clang
  - Анализ абстрактного синтаксического дерева программы
    - Поиск установленных шаблонов в коде программы
      - clang::RecursiveAstVisitor
    - Замена вызова функций по работе с памятью
      - clang::Rewriter

Анализ AST программы

```
for
{...
      buffer = (char*) malloc(sizeof(char));
        ...
      free(buffer);
...}
```



Анализ работы инструмента

- Инструмент был опробован на проекте “HipHacker’s Guide To The Galaxy”
  - Коллекция фундаментальных алгоритмов и структур данных
  - 7 из 40 алгоритмов библиотеки показали повышенную эффективность после оптимизации выделения памяти реализованным инструментом

Анализ работы инструмента

- stream_and_rank.c
  - уменьшение времени выполнения в среднем на 20 %

num_of_ints	Время работы программы **до**	Время работы программы **после**
300000	1,185 sec	0.795 sec
1000000	12.156 sec	10.192 sec
7000000	601.336 sec	474.023 sec

- merge_bottom_up.c
  - уменьшение времени выполения в среднем на 70 %

num_of_ints	Время работы программы **до**	Время работы программы **после**
10000	5.368 sec	0.557 sec
50000	67.732 sec	13.144 sec
150000	395.639 sec	180.799 sec

Результаты

- Написаны тестовые программы, проведены и проанализированы измерения времени работы программ
- Разработан прототип инструмента, оптимизирующего выделение памяти программы
- Были проведены эксперименты, демонстрирующие эффективность разработанного инструмента